Analysis and Enhancement of Conditional Random Fields Gene Mention Taggers in BioCreative II Challenge Evaluation

Yu-Ming Chang[1] , Cheng-Ju Kuo[1,2] , Han-Shen Huang[1] , Yu-Shi Lin[1,3] , Chun-Nan Hsu[1,*]

[1]Institute of Information Science, Academia Sinica, Taipei, Taiwan.
[2]Institute of Bioinformatics, National Yang-Ming University, Taipei,Taiwan.
[3]Department of Computer Science, National Taiwan University, Taipei, Taiwan.

Email: porter@iis.sinica.edu.tw; clarkkuo@iis.sinica.edu.tw; hanshen@iis.sinica.edu.tw; bathroom@iis.sinica.edu.tw; chunnan@iis.sinica.edu.tw;

*Corresponding author

Abstract

Background: Tagging gene and gene product mentions in scientific text is an important initial step of literature mining. In BioCreative 2 challenge, the conditional random fields model (CRF) was the most prevailing method in the gene mention task. In this paper, we analyze two best performing CRF-based systems in BioCreative 2. We examine their key claims and propose enhancement based on the analysis results.

Results: We implemented their systems in MALLET as specified in their report and in CRF++, a different CRF package, to empirically analyze their claims. We found that their feature set is effective for models trained by MALLET, but a smaller set works better for those by CRF++. We confirmed the effectiveness of pairing parentheses as a post processing step. We found that backward parsing is not always superior to forward parsing. The benefit of applying bidirectional parsing is the creation of a wider variety of complementary models. We elaborated the notion of divergent models by relating it to the difference of the increments of ture positives and false positives of the union model.

Conclusions: To further enhance the performance, we can integrate more models based on the elaborated notion of divergent models that we derived to minimize the number of models required.

Background

At present, scientific literature is still the largest and most reliable source of biomedical knowledge. A great deal of efforts have been devoted to literature mining in attempts to extract large volumes of biomedical facts, such as protein-protein interactions and disease-gene associations, from the literature. Curation of large-scale experimental data generated by high-throughput experimental methods also depends on literature mining. Literature mining usually takes many complex steps. Tagging gene and gene product mentions in scientific text is an important initial step. However, gene mention tagging is particularly difficult because authors rarely use standardized gene names and gene names naturally co-occur with other types that have similar morphology, and even similar context.

The second BioCreative challenge (BioCreative 2) [1] is a recent competition for biological literature mining systems. It took place in 2006 and followed by a workshop in April of 2007. This challenge consisted of a gene mention task, a gene normalization task and protein-protein interaction tasks. The gene mention task [2] evaluated how accurate a computer program can automatically tag gene names in sentences extracted from MEDLINE abstracts. Participants were given a tagged training corpus to develop their systems and an untagged test corpus to apply their systems for evaluation. The training corpus contains 15,000 sentences and the test corpus 5,000 sentences. Each run submitted by a participant was evaluated based on F-score, $F := \frac{2pr \cdot 100\%}{p+r}$, where p is precision and r is recall. A total of 21 participants submitted 3 runs to the challenge. The highest achieved F-score was 87.21.

In BioCreative 1 held in 2004, the conditional random fields model (CRF) [3] was applied in the gene metion tagging task and achieved high F-scores [4]. Of the 21 participants in BioCreative 2, 11 chose the CRF model. Apparently, CRF has become the most prevailing method in this task. In this paper, we analyze two CRF-based systems in BioCreative 2. One of them is Kuo et al.'s system [5], which is the best performing system based on CRF in BioCreative 2 (ranked 2nd). Its performance is not statistically significantly worse than any other system, and its performance is the best among all systems for a test corpus re-weighted to reflect the distribution of a random sentence extracted from MEDLINE. The other is Huang et al.'s system [6], which combines CRF with support vector machines (SVM) to achieve one of the best F-scores (ranked 3rd) in BioCreative 2. In fact, even the top performing system is not statistically significantly better than this system. The high performance of this combo system reconfirms a well-known strategy that combining multiple complementary models always improve the performance. Aside from their performance, both systems are interesting because they were built mostly on top of open source software packages for CRF and NLP, which makes it possible to duplicate their results.

Kuo et al.'s System

The key idea of Kuo et al.'s system [5] is to combine bidirectional parsing CRF models with a rich feature set. They used MALLET [7] to implement their CRF models to take advantage of its feature induction capability [8]. The CRF model is trained to label each token in an input sentence as one of `B` (beginning of a gene entity), `I` (in a gene entity), and `O` (outside a gene entity). Due to the special characteristics of name-entities of genes and gene products [9], a rich set of features is required to achieve satisfactory F-scores. Table 1 shows their feature set. Moreover, to include contextual information, they used -2 to 2 as the offsets to generate contextual features that apply to predicates including words, stemmed words and word morphology predicates at each position. To extract features, the Genia Tagger [10] was applied for stemming, tokenization and part-of-speech tagging. They modified the Genia Tagger slightly to tokenize words with a higher granularity. For example, punctuation symbols within words were segmented. They also applied a rule-based filter to clean up some easily fixed mistakes, such as entities with unpaired parentheses or square brackets.

To further improve its performance, they combined the tagging results of forward and backward parsing. In forward parsing, CRF reads and tags the input sentences from left to right, while in backward parsing, CRF reads and tags the input sentences from right to left. Figure 1 illustrates different parsing directions. We note that the training set and the "`B,I,O`" labels must both be reversed to train a backward parsing CRF model. That is, their backward parsing is equivalent to applying "`I,O,E`" labelling to reversed sentences [11] in named entity recognition. They tested the forward and backward parsing models and found that backward parsing constantly outperformed forward parsing in both recall and precision, but its reason is unclear. They assumed that some "signals" at the end of entities are more important to well demarcate boundaries of entities.

Finally, they applied a special method based on likelihood scores and dictionary-filtering, which used a dictionary-based filter to select entities from the union of the top ten tagging solutions obtained by MALLET's `n-best` option. In fact, the union of the top ten tagging solutions of bidirectional parsing achieved a nearly perfect recall at 98.10 for the final test, but with 13.87 precision. That is, nearly all true positives are in this union. They distilled real true positives from this union as follows.

1. Parse the input sentence in both directions to obtain the top ten solutions for each direction with their output scores;

2. Compute the intersection of bidirectional parsing and select the solution in the intersection that

minimizes the sum of its output scores;

3. For the other 18 solutions, select the labeled terms appearing in a dictionary with its length greater than three.

Step 2 is derived from the optimal model integration. Let x be a test sentence, y be a tagging, and m_i be a model where $i = 1, 2, \ldots$. The optimal integration of m_i's is to select y such that

$$y = \arg\max_y \prod_i p(y|x, m_i) = \arg\min_y -\sum_i \log p(y|x, m_i).$$

Next, they used approved gene symbols and aliases obtained from HUGO [12] as their dictionary for the final dictionary filtering.

Huang et al.'s System

Huang et al. [6] considered the gene mention tagging task as a classification problem and applied support vector machines (SVM) to solve it. They selected a large set of features as the input and trained two SVM models with different multiclass extension methods. They found that backward parsing constantly outperformed forward parsing regardless of the multiclass extension methods and obtained high precision rates, but recall rates were not as satisfactory. To enhance recall rates, their approach is to construct divergent but high performance models to cover different aspects of the feature space, and then combine them into an ensemble. They also applied union and intersection to combine the outputs of SVM models with that of a CRF model, which was trained with the same feature set, and successfully enhanced recall rates without degrading too much precision. They chose Yet Another Multipurpose Chunk Annotator (YamCha) [11, 13] to build their SVM models because it is tuned for name entity chunking tasks. They designed their features based on the experience and previous works on named entity recognition [4, 14]. Table 3 shows the set of features. There are a total of 617,515 features in the feature set.

They also used an inside/outside representation for gene mention tagging with B, I, and O class labels. Since SVM is an intrinsic binary classifier, extensions must be made to handle multiclass problem. They applied two popular methods to extend a binary classifier to multiclass: one vs. all and one vs. one. They also trained a conditional random field (CRF) model to increase the divergence of the ensemble.

Table 4 shows the final test results of this model, as well as the final results of the unions of CRF with the two SVM models. The results show that the simple ensemble model significantly enhanced recall, with all recall results ranked in the top quartile, while precision results dropped slightly. All F-score results were ranked in the top quartile among 21 participants, too.

Results and Discussion

In this section, we present the results of our investigation of the key claims of these systems with discussions. We developed a JAVA feature extractor for MALLET to duplicate their results. To investigate the portability, we also applied CRF++ [15], another free package for CRF training. In addition, we applied CTJPGIS, a new algorithm for CRF training to compare its performance with L-BFGS [16], the de facto standard algorithm for CRF training. The derivation of CTJPGIS is presented in method section. Though our focus is mainly on CRF, we also include the YamCha implementation of SVM in our investigation.

Feature Selection

To investigate the impact of feature selection, our plan was to remove a subset of features corresponding to a feature type and observe its impact to the performance. This is expensive because training CRF takes time. We duplicated Kuo et al.'s feature set as shown in Table 1 for models trained by MALLET. This set will be denoted as F1. We also duplicated Huang et al.'s feature set as shown in Table 3. This set will be denoted as F2. The difference between these two sets of features includes the N-grams in F1 and the prefix and suffix features in F2. We changed Huang et al's set and slightly to improve the performance of SVM by removing orthographic features. We used this new set of features for models trained by CRF++ because YamCha and CRF++ share the same input format of the features. We compared the performance of the CRF++ tagger with F2 including and not including orthographic features. The performance was improved slighly without orthographic features. Moreover, the F-score 87.12 by CRF++ is already higher than Kuo et al.'s system and this is achieved by a single CRF model without model integration.
We tried our MALLET model to see if the removal of orthographic features also helps. It turns out that the F-score drops significantly by 4 percentage points. The results are shown in Table 5. We removed other subsets from the feature set for MALLET and observed similar performance degradation. Therefore, F1, as given in Table 1, is effective for models trained by MALLET, but a smaller set works better for those by CRF++, and the selection of best features depends on the CRF package we used. In the following discussion, models trained by MALLET will use F1 as the feature set while models trained by CRF++ will use F2 excluding orthographic features F2−O, which will also be used by SVM models trained by YamCha.

Post Processing

We implemented the post processing step that resolves problems caused by unpaired parenthesis. For example, suppose the input sentence is

> ... implicated the NIMA (never in mitosis, gene A)-related kinase-6 (NEK6) ...

but the model tags "gene A)-related kinase-6" as a gene mention. This tagged entity is obviously incorrect because it includes only a right parenthesis. The post processing program will first find the left parenthesis in the sentence, and then search if there is a stop word or parenthesis at the left side of the left parenthesis. In this example, the second token "the" conforms to the condition. At last, the program will extend the tagged entity to the token right after the stop word "the" and output the extended string "NIMA (never in mitosis, gene A)-related kinase-6" as the tagged entity, which is a correct tagging. When the tagged entity contains only a left parenthesis, reverse the search direction for parenthesis and stop word.
Table 6 shows the improvement of the post processing for two different models. Though the improvement is less than a percentage point in F-score, for all models, both precision and recall are improved, suggesting no trade-off and the effectiveness of pairing parentheses as a post processing step.

Backward and Forward Parsing

The most prominent feature of Kuo et al. and Huang et al's systems is the use of backward parsing. However, the superiority of backward parsing is also the most speculative. Intuitively, it can be explained that some "signals" at the end of gene entities enable the system to recognize them more accurately. For example,

> ... zinc finger and BTB domain-containing *protein* 39 ...

The words highlighted at the end appear to be the "signal." The empirical results presented in their report appeared to be evidential, but it is not clear whether the claim is applicable to other data sets or whether the claim is only specific to the choice of feature sets.
We started by investigating the difference of the trained models of the forward and backward parsing by applying MALLET with the same feature set used in Kuo et al.'s system. We counted the percentage differences between the forward and backward parsing models of nonzero predicates and errors made at each type of class transition positions. Nonzero predicates are important because they are the only predicates that will be considered by the CRF model when it tags a sentence. The result is shown in Table 7, which shows that nonzero predicates are different between forward and backward parsing for

transitions involving B's, but no significant difference in errors made at those transitions. There appears no correlation between the differences and the errors made.

Then we applied CRF++ to see if the superiority of backward parsing is portable to another CRF package. The feature set used by the CRF++ tagger is slightly different from the MALLET version, as discussed in the sub-section about feature selection. In a nutshell, we compared forward and backward parsing for the same BioCreative 2 data set but used different CRF packages with different sets of features. The result is shown in Table 8. It turns out that as reported in [5], the MALLET tagger performs better applying backward parsing than forward parsing by 0.47 percentage points in F-score, but on the contrary to Kuo et al.'s results, our CRF++ tagger performs worse applying backward parsing by 0.05 percentage points. The difference column in Table 8 shows that the difference between forward and backward parsing models for MALLET is ten times as many as that for CRF++, which leads to significant improvement of precision by intersection of forward and backward parsing models for MALLET but tiny improvement for CRF++. We also compared the log-likelihood scores obtained by MALLET and CRF++ for both parsing directions and found that the log-likelihood scores are almost the same for CRF++ but different for MALLET.

We conclude that backward parsing is not always superior to forward parsing. The benefit of applying bidirectional parsing is the creation of a wider variety of complementary models.

Model Integration

Another prominent feature of Kuo et al. and Huang et al.'s systems is that combining divergent but high performance models always improve the performance. To create divergent but high performance models, we implemented the following four models:

- The first one is the intersection of forward and backward parsing models trained by MALLET with its default training algorithm L-BFGS, denoted by `MalletL-BFGSint`. Intersection was applied to boost its precision. The feature set used is F1.
- The second one is a forward parsing model trained by CRF++, denoted by `CRF++L-BFGS`, also using its default training algorithm L-BFGS.
- We altered the training algorithm of CRF++ from L-BFGS to CTJPGIS to train the third model, called `CRF++CTJPGIS`. This new algorithm is introduced to create a wider variety of models. Since CTJPGIS is derived quite differently from L-BFGS [16], their search paths to the optimum are quite

different, too. Therefore, CTJPGIS can be applied to create a model to complement the model trained by L-BFGS. Both CRF++ models use F2−O as their feature set.

- The forth model is the forward parsing SVM model trained by `YamCha`. The input feature set is also F2−O as discussed in feature selection sub-section.

Tabel 9 shows the precision, recall and F-score of the four models and their unions. The results show that with a similar set of features, we can duplicate Kuo et al.'s performance by CRF models trained by a different software package. In fact, both models trained by CRF++ outperform Kuo et al.'s best system, and their union outperforms the rank 1 system in BioCreative 2, which achieved a F-score of 87.21. Our best performing system is the union of two models trained by L-BFGS, achieving a F-score of 87.67. We note that no external data source other than the training corpus provided by BioCreative 2 was used and only pairs of integrated models was required to achieve this result.

Other notable results include that the difference of F-scores between the YamCha model and two CRF++ model is surprisingly large even though they share the same set of features, and that the unions of SVM and CRF models perform not as well as the unions of CRF model pairs. Intuitively, the variance between the tagging results of a SVM model and a CRF model is supposed to be larger than that between CRF models and therefore the union between SVM and CRF models is supposed to complement each other's false negatives and perform better. But the results show differently. An explanation is that the performance of the SVM model is not as good as its CRF counterparts. But the performance of `MalletL-BFGSint` is not as good neither, though its precision is very high. We seeked the answer by examining the difference of true positives and false positives before and after applying union to each pair of models. Compared to the F-score results in Table 9, we can see that the more increment of TP and the less increment of FP bring a higher F-score. Figure 2 illustrates this finding with bar charts, which show that the larger the difference between the increments of TP and FP, the larger the gain of F-score by union. If the difference is negative, then union may degrade the performance when individual tagger's performance is not sufficiently high. This explains why the union of MalletL-BFGSint and CRF++L-BFGS performs much better than the unions of YamCha and other CRF models.

Conclusions

We have analyzed the key claims of the two best performing CRF-based gene mention taggers in BioCreative 2 and proposed simple enhancement that performs better than any system in BioCreative 2.

We showed that the set of features used by Kuo et al. is effective for MALLET, but when applied to CRF++, it is not as effective as a set with orthographic features removed. Then we showed that balancing parentheses as a post-processing step always improves the performance. We analyzed a prominent claim that backward parsing models is superior for gene mention tagging. We found that it may apply to MALLET models but not apply to CRF++. No significant difference between backward and forward parsing was observed for CRF++ models. It is not clear why MALLET and CRF++ respond differently with regards to feature selection and parsing direction. In theory, they implement the same L-BFGS algorithm for the same CRF model. We suspect that approximation in model inference may play a role in their different behavior. However, this cannot be elucidated unless we actually trace their code to figure out the real cause. Finally, we confirmed that integrating divergent models improves the performance. We elaborated the notion of divergent models by relating it to the difference of the increments of ture positives and false positives of the union model.

To further enhance the performance, we can integrate more models based on the elaborated notion of divergent models that we derived. A grand ensembl of all participating systems in BioCreative 2 achieves a F-score of 90.6 [2]. Our heuristic can be used to minimize the number of models required.

Methods

CTJPGIS

CTJPGIS is the abbreviation of "*the componentwise triple jump method for penalized generalized iterative scaling*." CTJPGIS is derived from the generalized iterative scaling (GIS) method [17], which is a classical method to train exponential probabilistic models. However, GIS usually converges slowly, especially when applied to train a CRF model for large-scale gene mention tagging tasks.

Since GIS can also be considered as fixed-point iteration [18], we can apply the triple jump extrapolation method to speed up its convergence. The triple jump method is an approximation of Aitken's acceleration for fixed-point iteration methods. It has been successfully applied to the EM algorithm [19–22]. The idea is to estimate the extrapolation rate by considering the previous two consecutive estimates of the parameter vectors. The triple jump extrapolation method can effectively accelerate the EM algorithm by substantially reducing the number of iterations required for the EM algorithm to converge. Though the triple jump method, as all variants of Aitken's acceleration, may not monotonically increase the likelihood, we can apply the idea proposed by [23] to resolve the issue. The idea is to discard the extrapolation if it fails to improve the likelihood and use the estimate obtained without the extrapolation. In this way, convergence

can be guaranteed [20]. CTJPGIS runs as fast as L-BFGS, therefore, we can create many models efficiently.

References

1. Hirschman L, Krallinger M, Valencia A: *Proceedings of the second BioCreative challenge evaluation workshop*. CINO Centro Nacional de Investigaciones Oncologicas 2007.
2. Wilbur J, Smith L, Tanabe L: **BioCreative 2. Gene Mention Task**. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop* 2007:7–16.
3. Lafferty J, McCallum A, Pereira F: **Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data**. In *Proceedings of 18th International Conference on Machine Learning (ICML '03)* 2001:282–289.
4. McDonald R, Pereira F: **Identifying gene and protein mentions in text using conditional random fields**. *BMC Bioinformatics* 2005, **6**:S6.
5. Kuo CJ, Chang YM, Huang HS, Lin KT, Yang BH, Lin YS, Hsu CN, Chung IF: **Rich Feature Set, Unification of Bidirectional Parsing and Dictionary Filtering for High F-Score Gene Mention Tagging**. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop* 2007:105–107.
6. Huang HS, Lin YS, Lin KT, Kuo CJ, Chang YM, , Yang BH, Hsu CN, Chung IF: **High-Recall Gene Mention Recognition by Unification of Multiple Backward Parsing Models**. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop* 2007:109–111.
7. McCallum AK: **MALLET: A Machine Learning for Language Toolkit** 2002. [Http://mallet.cs.umass.edu].
8. McCallum A: **Efficiently inducing features of conditional random fields**. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)* 2003[citeseer.ist.psu.edu/mccallum03efficiently.html].
9. Zhou GD: **Recognizing names in biomedical texts using mutual information independence model and SVM plus sigmoid.** *International Journal of Medical Informatics* 2006, **75**:456–467.
10. Tsuruoka Y, Tateishi Y, Kim JD, Ohta T, McNaught J, Ananiadou S, Tsujii J: **Developing a Robust Part-of-Speech Tagger for Biomedical Text**. In *Advances in Informatics - 10th Panhellenic Conference on Informatics* 2005:382–392.
11. Kudo T, Matsumoto Y: **Chunking with support vector machines** 2001, [citeseer.ist.psu.edu/kudo01chunking.html].
12. Eyre TA, Ducluzeau F, Sneddon TP, Povey S, Bruford EA, Lush MJ: **The HUGO Gene Nomenclature Database, 2006 updates**. *Nucleic Acids Research* 2006, **34**:D319–D321.
13. Kudo T: **YamCha: Yet Another Multipurpose CHunk Annotator** 2001. [Http://chasen.org/ taku/software/yamcha/].
14. Mitsumori T, Fation S, Murata M, Doi K, Doi H: **Gene/protein name recognition based on support vector machine using dictionary as features**. *BMC Bioinformatics* 2005, **6**:S8.
15. Kudo T: **CRF++: Yet Another CRF toolkit** 2005. [Http://crfpp.sourceforge.net/].
16. Nocedal J, Wright SJ: *Numerical Optimization*. Springer 1999.
17. Darroch JN, Ratcliff D: **Generalized iterative scaling for log-linear models**. *The Annals of Mathematical Statistics* 1972, **43**(5):1470–1480.
18. Burden RL, Faires D: *Numerical Analysis*. PWS-KENT Pub Co. 1988.
19. Hsu CN, Huang HS, Yang BH: **Global and Componentwise Extrapolation for Accelerating Data Mining from Large Incomplete Data Set with the EM Algorithm. To appear in *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM '06).*** 2006.
20. Huang HS, Yang BH, Hsu CN: **Triple-Jump Acceleration for the EM Algorithm**. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM '05)* 2005:649–652.
21. Hesterberg T: **Staggered Aitken Acceleration for EM**. In *Proceedings of the Statistical Computing Section of the American Statistical Association*, Minneapolis, Minnesota, USA 2005.
22. Schafer JL: *Analysis of Incomplete Multivariate Data*. London: Chapman and Hall / CRC Press 1997.
23. Salakhutdinov R, Roweis S: **Adaptive overrelaxed bound optimization methods**. In *Proceedings of the Twentieth International Conference on Machine Learning* 2003:664–671.

Figures and Tables

Table 1: Features of Kuo et al.'s system.

Feature	Example	Feature	Example	Feature	Example
Word	proteins	Hyphen	-	Nucleoside	Thymine
StemmedWord	protein	BackSlash	/	Nucleotide	ATP
PartOfSpeech	NN	OpenSqure	[	Roman	I, II, XI
InitCap	Kinase	CloseSqure	]	MorphologyTypeI	p53→p*
EndCap	kappaB	Colon	:	MorphologyTypeII	p53→a1
AllCaps	SOX	SemiColon	;	MorphologyTypeIII	GnRH→AaAA
LowerCase	interlukin	Percent	%	WordLength	1, 2, 3-5, 6+
MixCase	RalGDS	OpenParen	(	N-grams(2-4)	p53→{p5, 53}
SingleCap	kDa	CloseParen	)	ATCGUsequece	ATCGU
TwoCap	IL	Comma	,	Greek	alpha
ThreeCap	CSF	FullStop	.	NucleicAcid	cDNA
MoreCap	RESULT	Apostrophe	’	AminoAcidLong	tyrosine
SingleDigit	1	QuotationMark	”	AminoAcidShort	Ser
TwoDigit	22	Star	*	AminoAcid+Position	Ser150
FourDigit	1983	Equal	=		
MoreDigit	513256	Plus	+		

Table 2: System performance of submitted runs by Kuo et al.

System	Precision	Recall	F-Measure
Backward	89.30	83.83	86.48
Union	86.10	87.08	86.58
Top Ten + Dictionary	89.30	84.49	**86.83**

Table 3: Types of features of Huang et al.'s system and their possible values.

Feature	Value
word	all words in the training data
POS	part-of-speech tagging by GENIA tagger
orthographic	same as the features from InitCap to Plus in Table 1
vowel	a,e,i,o,u
length	1,2,3∼5,≥6
morphological I	replacing digits with a "*"(e.g., Abc123→Abc*)
morphological II	replacing each letter and digit with a morphological symbol (e.g., AbcD123→AaaA111)
prefix	1∼6 gram of the starting letters of the token
suffix	1∼6 gram of the ending letters of the token
preceding class	class labels of the two preceding tokens

Table 4: Final results of Huang et al. systems.

Run	Ensemble	Performance
1	M1∪M3	P:83.27(3) R:89.34(1) F:86.20(1)
2	M2∪M3	P:82.98(3) R:89.58(1) F:86.15(1)
3	(M1∩M2)∪M3	P:84.93(3) R:88.28(1) F:86.57(1)

The number in the parentheses is the quartile among 21 participants.
M1 = SVM + one vs. all, M2 = SVM + one vs. one, M3 = CRF.

Table 5: Comparison of different features. F1 is the feature set given in Table 1, F2 is the feature set given in Table 3, and O denotes orthographic features. "−" denotes set difference.

Model		F1	F1−O	Model		F2	F2−O
MalletL-BFGS forward	Precision	88.88	79.80	CRF++L-BFGS	Precision	90.16	90.15
	Recall	83.57	84.64		Recall	84.13	84.28
	F-score	**86.14**	82.15		F-score	87.04	**87.12**

Table 6: Improvment by post processing for all models.

Model		Without postprocess	With postprocess
MalletL-BFGSint	Precision	91.93	92.12
	Recall	75.36	75.69
	F-score	82.82	83.10
CRF++L-BFGS	Precision	89.85	90.15
	Recall	83.87	84.28
	F-score	86.76	87.12

Table 7: Percentage differences between forward and backward parsing models of nonzero predicates and errors made at each type of class transition positions

$\Delta\%$	O,O	O,B	B,O	B,B	B,I	I,O	I,B	I,I	Total
Nonzero predicates	-0.1	**25.6**	**-37.0**	-6.7	**-18.2**	-8.3	**13.4**	3.5	-0.03
Errors	0.4	1.5	8.8	**21.7**	-3.1	-3.7	1.9	-4.0	-0.17

Table 8: Comparison between forward and backward parsing.

Model		Forward	Backward	Intersection	Difference
MalletL-BFGS	TP	5291	5321	4792	1028
	FP	662	635	410	477
	Precision	88.88	89.34	92.12	-
	Recall	83.57	84.05	75.69	-
	F-score	**86.14**	**86.61**	83.10	0.47
CRF++L-BFGS	TP	5336	5326	5281	100
	FP	583	577	557	46
	Precision	90.15	90.22	90.46	-
	Recall	84.28	84.13	83.41	-
	F-score	**87.12**	**87.07**	86.79	0.05

Table 9: Precision, Recall and F-score of each individual model (diagonal elements) and the union of each pair of models (off-diagnoal elements).

Model		MalletL-BFGSint	CRF++L-BFGS	CRF++CTJPGIS	YamCha
MalletL-BFGSint	Precision	92.11	**88.67**	88.65	84.98
	Recall	75.69	**86.68**	86.40	87.03
	F-score	83.10	**87.67**	87.51	85.99
CRF++L-BFGS	Precision		90.15	88.21	84.16
	Recall		84.28	86.59	88.01
	F-score		87.12	87.39	86.05
CRF++CTJPGIS	Precision			90.60	84.39
	Recall			82.96	87.73
	F-score			86.61	86.03
YamCha	Precision				86.96
	Recall				80.70
	F-score				83.71

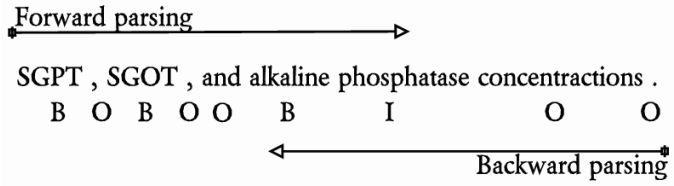

Figure 1: An example of sentence forward parsing and backward parsing: In forward parsing, the tagger reads a sentence from left to right as its ordinary order. But in backward parsing, the tagger reads a sentence from right to left. That is, in its reversed order.

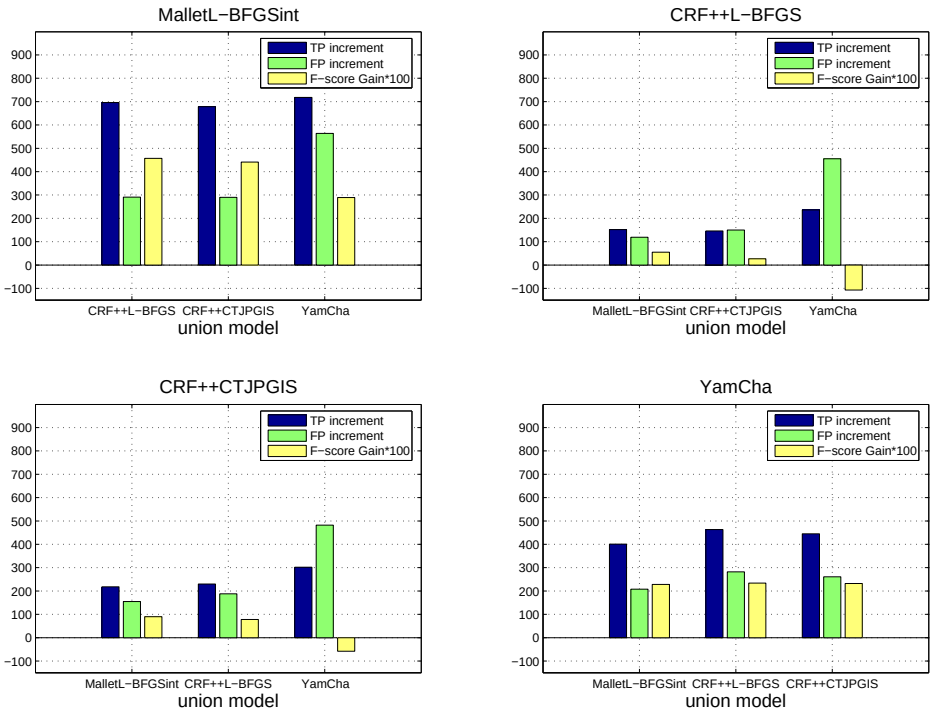

Figure 2: Relation between the increments of true positive and false positive and the gains of F-score of union models.